

Using Quadrilaterals to Compute the Shortest Path

Extended Abstract

Newton Campbell Jr.
Nova Southeastern University
nc607@nova.edu *

March 4, 2016

Abstract

We introduce a new heuristic for the A* algorithm that references a data structure of size $\theta(|L|^2 + |V|)$, where L represents a set of strategically chosen landmark vertices and V the set of vertices in the graph. This heuristic's benefits are permitted by a new approach for computing landmark-based lower bounds, in which each landmark stores only the distances between it and vertices within its graph partition. During search queries, a geometric inequality based on distance information for multiple landmarks is used to establish a lower bound for the search. In comparison to previous landmark-based algorithms, this process significantly reduces the amount of preprocessed distance information that needs to be stored (typically $\theta(|L| \cdot |V|)$) while also granting a constant computational cost for each vertex visited during a shortest path query. Further, for graphs with non-overlapping partitions, preprocessing this data structure requires time complexity equivalent to one Dijkstra's shortest path tree computation on the graph.

We characterize the behavior of this new heuristic based on a dual landmark configuration that leverages quadrilateral inequalities to identify the lower bound for shortest path. Using this approach, we demonstrate both the utility and detriments of using polygon inequalities aside from the triangle inequality to establish lower bounds for shortest path queries. While this new heuristic does not dominate previous heuristics based on triangle inequalities, the inverse is true, as well. Further, we demonstrate that an A* heuristic function does not necessarily outperform another heuristic that it dominates. In comparison to other landmark methods, the new heuristic maintains a larger average search space while commonly decreasing the number of computed arithmetic operations. The new heuristic can significantly outperform previous methods, particularly in graphs with larger path lengths. The characterization of the use of these inequalities for bounding offers insight into its applications in other theoretical spaces.

*This research was conducted at Nova Southeastern University Graduate School of Computer and Information Sciences in partial fulfillment for the requirements for the degree of the Doctor of Philosophy (Ph.D.) in Computer Science. Tuition assistance for the program was provided by the student's employer, Raytheon BBN Technologies.

1 Introduction

Since Euler’s early analysis of the map of Knigsberg [5], a large body of work has been devoted to pathfinding problems such as the point-to-point shortest path problem. As researchers find more use for graph theory in the storage, retrieval, and analysis of big data, extremely fast solutions to problems such as the shortest path problem are in great demand. Not even Dijkstra’s or the A* algorithm can solve the problem for massive datasets without an exponential increase in their requirements for computational time and space. Consequently, modern research focuses on computing and storing a data structure derived from the graph structure prior to allowing it to be queried for shortest path. This data structure is used to guide, narrow, or inform the search such that arbitrary queries can be performed significantly faster on graphs that represent huge data corpuses. Modern approaches typically exploit mathematical estimations [3, 2, 7, 13], large-scale storage [4, 9, 17, 16], artificial intelligence algorithms [1, 19, 20, 21], and combinations of preprocessing algorithms [15].

We introduce an approach for shortest path estimation that was derived from ALT[7], a class of algorithms that leverage \underline{A}^* search, a set of strategically chosen vertices called landmarks, and the triangle inequality to obtain lower bounds for shortest path queries. ALT works as follows: prior to search, the distances between a chosen set of vertices in a graph and all other vertices are computed and stored in a data structure. For a set of landmarks \mathbf{L} , graph vertex set $|V|$, and graph edge set $|E|$, this data structure is of size $\theta(|\mathbf{L}| \cdot |V|)$ and is computed in $\theta(|\mathbf{L}| \cdot (|E| + |V| \log |V|))$ time. During search queries, source and target information contained in this data structure is leveraged to identify a triangle in the graph, allowing the triangle inequality to be established as a lower bound for the shortest path. Our approach, known as ALP, is a class of algorithms that leverage \underline{A}^* search, landmarks, and (other) polygon inequalities. In this paper, demonstrate that ALP leverages a data structure of size $\theta(|\mathbf{L}|^2 + |V|)$ as opposed to ALT’s previous $\theta(|\mathbf{L}| \cdot |V|)$. This data structure is permitted by a new embedding process for identifying distances only within a graph partition. The scenarios and configurations in which each heuristic outperforms the other are enumerated. Finally, we highlight the ability for Dijkstra’s algorithm to outperform both ALT and ALP.

1.1 Previous Work on Shortest Path Preprocessing

Significant work has been done in preemptively analyzing graphs to store information that can assist in solving the point-to-point shortest path (PPSP) problem. Geometric Containers [18] algorithms relies on the concept of edge labeling, where preprocessing attaches a label to each edge in a graph that represents all nodes to which a shortest path starts with the individual edge. This means that for a graph $G = (V, E)$, Geometric Containers maintain a linear space requirement of $\theta(|E|)$. However, the preprocessing step requires a single source shortest path search from every node, requiring $\theta(|V| \cdot (|E| + |V| \log |V|))$ preprocessing time. Arc Flags algorithms [12] require an input graph that is partitioned such that a flag is computed for each edge within a partition, or region, which indicates whether the edge is on a shortest path to any node in that partition. It is similar to the Geometric Containers in that it considers only the edges whose flag correspond to a specific region. This algorithm realizes a high preprocessing time, as one shortest path search from every border node of a region is required. For B border nodes and k partitions, Arc Flags preprocessing maintains a time complexity of $\theta(B \cdot |E| + |V| \log |V|)$ with a space requirement of $\theta(k \cdot |E|)$.

The precomputed cluster distances (PCD) algorithm [14] was designed with the intention of reducing the space requirements of preprocessing algorithms such as ALT [7] (detailed in the next section). PCD leverages the distances between graph clusters to compute the heuristic for A*. The algorithm has demonstrated practical performance benefits over the ALT, Arc Flags and Geometric Containers preprocessing algorithms. It is often noted that the space complexity for PCD is $\theta(k^2 + B)$ compared to ALT’s $\theta(|\mathbf{L}| \cdot |V|)$, where k is equal to the number of graph clusters, B is equal to the number of border nodes for clusters, and L is the set of chosen landmark vertices. However, since the actual clustering information is stored for PCD, as well,

the actual space complexity is $\theta(k^2 + B + |V|)$, as information about which cluster every vertex belongs to needs to be maintained by a data structure in order to be referenced. PCD still boasts a higher experimental average speedup for PPSP queries compared to ALT and has a smaller preprocessing time complexity.

Reach-based pruning is another method for speeding up shortest-path queries such as Dijkstra’s algorithm. Reach is a centrality measure that identifies how central a vertex is on a shortest path [10]. The latest methods for preprocessing the reach for all nodes requires $O(|V|^2 \log V)$ time and $O(|V|)$ storage. Reach-based pruning is often combined with ALT to perform REAL, a method that stores landmark distances of only high reach[6].

1.2 Outline

We begin the technical part of this paper in Section 2 with a review of basic graph theory, the ALT class of algorithms, and embedding methods for ALT. In Section 3, we describe the ALP approach in detail, with a focus on identifying quadrilaterals in the graph. In Section 4, we discuss a new method of forming a landmark-based preprocessing data structure called distributed embedding. In Section 5, we discuss how different heuristics should be compared for A* to more closely map heuristic metrics to computational performance. Finally, Section 6 describes how the research can be carried forward.

2 Using Landmarks To Calculate Shortest Path

2.1 Graph Notation

Throughout the rest of this paper, a common set of graph theoretical definitions, concepts, and notations will be used. Let $G = (V, E)$ be an undirected graph, where V is the set of vertices in G and $E \subseteq V \times V$ is the set of edges in G , with $n = |V|$ and $m = |E|$. For any edge $e \in E$, let $w(e)$ be the positive real *weight* of e . In an *unweighted* graph, for every edge $e \in E$, $w(e) = 1$. In a *weighted* graph, $w(e)$ is subject to the graph’s application. A *finite* graph is one in which $|V| \neq \infty$ and $|E| \neq \infty$. If an edge $e \in E$ connects two vertices $v_i, v_j \in V$, v_i is called the *neighbor* of v_j and v_j the *neighbor* of v_i . The vertices v_i and v_j are also said to be *adjacent* to each other and *incident* to their shared edge e . A graph $H = (V(H), E(H))$ is a *subgraph* of G if $V(H) \in V$ and $E(H) \in E$, with edges of $E(H)$ incident to only the vertices in $V(H)$. A *spanning subgraph* H of G is a subgraph in which $V(H) = V$. An *induced subgraph* H of G is a graph whose set of vertices are a subset of the vertices of G and whose set of edges have only endpoints in $V(H)$. A graph *cluster*, or *community* is a collection of vertices in a graph such that the vertices assigned to a particular community are similar or connected by some predefined criteria.

A sequence (v_0, \dots, v_{k-1}) , $k \geq 1$, of vertices of $G = (V, E)$ is known as a *path* from v_0 to v_{k-1} if there is an edge $(v_i, v_{i+1}) \in E$ for every $0 \leq i < k$. A path is denoted as $P(v_0, v_{k-1}) = v_0, \dots, v_{k-1}$. A path P is also a subgraph of G . The *path length* of P is the number of edges (i.e., $k - 1$) on the path $P(v_0, v_{k-1})$, denoted as $d(v_0, v_{k-1})$ or $d(P)$. If a path is the shortest path, its path length is referred to as the *distance* from v_0 to v_{k-1} . The *path weight* of P is the sum of the weights of the path edges, denoted as $w(P)$ or $w(v_0, v_{k-1})$. If, for every pair of vertices $v_i, v_j \in V$, there exists a path from v_i to v_j , the graph is called *connected*. An acyclic, connected, spanning subgraph of G is called a *spanning tree* of G . In this paper, the algorithms focus on undirected, unweighted, finite, connected graphs.

2.2 ALT (A*, Landmarks, and Triangle Inequalities)

In ALT, a shortest path query uses a computed distance estimate, derived from the triangle inequality, to guide the search. Let $L \in V$ be the set of landmarks with distance $d(v, l_i)$ stored for all vertices $v \in V$ and

any landmark $l_i \in \mathbf{L}$, $1 \leq i \leq |\mathbf{L}|$. Using the shortest path distances for graph $G = (V, E)$, this inequality yields two important equations for any three vertices $s, t, l_i \in V$:

$$d(s, t) \leq d(s, l_i) + d(l_i, t) \quad (1)$$

$$d(s, t) \geq |d(s, l_i) - d(l_i, t)| \quad (2)$$

Based on these inequalities, ALT works as follows: In a preprocessing step, the Dijkstra's shortest path tree (SPT) algorithm is used to compute and store the distances to each landmark in \mathbf{L} from all other vertices in V . Then, during shortest path distance queries, the triangle inequality is used as follows: let $\pi_t^L(v)$ denote the heuristic function based on landmarks used for the A* algorithm. The following equation represents the ALT heuristic function when visiting vertex $v \in V$ on the way to a target vertex t :

$$\pi_t^L(v) = \max_{l_i \in \mathbf{L}} |d(v, l_i) - d(t, l_i)| \quad (3)$$

2.3 Embedding Methods

Briefly, we review the most common *embedding methods*, or *landmark selection techniques*, for ALT. *Random* landmark selection [7] identifies and tests k vertices at random to serve as landmarks (where k is typically a parameter). In terms of lower bounds, random landmark selection demonstrate better performance than any of the following methods of landmark selection [8]. *Farthest* landmark selection [7] identifies vertices that maximize path weight between each other. A later algorithm for farthest landmark selection was created to maximize path distance instead of path weight [8]. This biases farthest selection to choose separate, dense regions of the graph to place landmarks in. *Planar* landmark selection [7] uses graph layout information to divide a graph into sectors and then executes *farthest* landmark selection. *Avoid* landmark selection, a commonly used and modified landmark selection algorithm, computes the SPT T_r , rooted at a random vertex $r \in V$ [8] and strategically identifies leaves of the tree to serve as landmarks. This approach "avoids" existing landmarks to improve coverage of landmarks over the graph.

3 A*, Landmarks, and Polygon Inequalities

3.1 Quadrilateral-Based Inequalities in Graphs

ALT computes shortest path trees (SPTs) from a selected set of landmarks and uses the triangle inequality at query time to establish a lower bound for A* [7]. Here, we demonstrate that these heuristics can be derived from other geometric inequalities by identifying other types of polygons in a graph and setting the heuristic values for A* equal to the maximum lower bound. The ALT algorithm is the base case for such a hypothesis. The use of two landmarks, as seen in this paper, provides an inductive step for the proof of the hypothesis. To show this, first, we describe how to form a triangle in a graph to establish the triangle inequality as a lower bound. This proof was derived from the reverse triangle inequality proof for a metric space (detailed in full paper).

For a simple graph G with vertices $A, B, C \in V$, the shortest path distances between each vertex allow the graph to form a metric space. Therefore, for the distances between vertices $A, B, C \in V$, the following triangle inequalities hold:

$$d(A, B) \leq d(A, C) + d(C, B) \quad (4)$$

$$d(B, C) \leq d(B, A) + d(A, C) \quad (5)$$

Both of these inequalities apply to the three vertices in G . The reverse triangle inequality, which is used as a lower bound for A* in ALT, is derived from these inequalities. ALT uses this reverse triangle inequality

#	Statements			Reasons
	A	B	C	
1.	$d(A, B) \leq d(B, C) + d(C, D) + d(A, D)$	$d(B, C) \leq d(A, B) + d(C, D) + d(A, D)$	$d(C, D) \leq d(A, B) + d(B, C) + d(A, D)$	Quadrilateral Inequality (Given)
2.	$d(A, B) - d(B, C) - d(C, D) \leq d(A, D)$	$d(B, C) - d(A, B) - d(C, D) \leq d(A, D)$	$d(C, D) - d(B, C) - d(A, B) \leq d(A, D)$	Subtraction on both sides #1
3.	$d(A, B) - d(C, D) - d(B, C) \leq d(A, D)$	$d(B, C) - d(C, D) - d(A, B) \leq d(A, D)$	$d(C, D) - d(A, B) - d(B, C) \leq d(A, D)$	Subtraction on both sides #1
4.	$ d(A, B) - d(B, C) - d(C, D) \leq d(A, D)$			Absolute Value Definition (#2A/2B)
5.	$ d(B, C) - d(C, D) - d(A, B) \leq d(A, D)$			Absolute Value Definition (#2C/3B)
6.	$ d(A, B) - d(C, D) - d(B, C) \leq d(A, D)$			Absolute Value Definition (#3A/3C)

Figure 1: Derivation of the Reverse Quadrilateral Inequality in Simple, Connected Graphs

to create a heuristic that estimates the distance between vertices A and C by setting vertex B equal to a landmark l such that

$$|d(A, l) - d(l, C)| \leq d(A, C) \quad (6)$$

By computing and storing the values $d(A, l)$ and $d(l, C)$ before performing any PPSP queries, this lower bound is then used as a heuristic to the A* algorithm. Because it is the lower bound, it will never overestimate the distance between vertices A and C .

For a quadrilateral, the lower bound of one of its sides can also be calculated using the other three sides. This reverse quadrilateral inequality can also be used to establish the lower bounds for the shortest path of a graph. For a graph G with vertices $A, B, C, D \in V$, Figure 1 demonstrates how we derived the following reverse quadrilateral inequalities based on upper bound estimates:

$$|d(A, B) - d(B, C)| - d(C, D) \leq d(A, D) \quad (7)$$

$$|d(C, D) - d(A, B)| - d(B, C) \leq d(A, D) \quad (8)$$

$$|d(B, C) - d(C, D)| - d(A, B) \leq d(A, D) \quad (9)$$

A potential problem with these inequalities is that they have ability to generate negative lower bound estimates, which is useless for a nonnegative distance metric. For utility, when attempting to estimate the lower bounds of a quadrilateral, other inequalities should be considered such that the highest possible lower bound can be used. In this paper, we use Ptolemy's inequality [11] to demonstrate an example of this. Ptolemy's inequality can be applied to the graph quadrilateral as follows to yield lower bounds for the distance between A and D . We begin with the original inequality:

$$d(A, C) \times d(B, D) \leq d(A, B) \times d(C, D) + d(B, C) \times d(A, D) \quad (10)$$

Then to estimate the distance between A and D , using simple algebra,

$$\frac{d(A, C) \times d(B, D) - d(A, B) \times d(C, D)}{d(B, C)} \leq d(A, D) \quad (11)$$

1	$\pi_t^{DL}(v, 1) = d(v, l_1) - d(l_1, l_2) - d(l_2, t)$
2	$\pi_t^{DL}(v, 2) = d(v, l_1) - d(l_2, t) - d(l_1, l_2)$
3	$\pi_t^{DL}(v, 3) = d(l_1, l_2) - d(l_2, t) - d(v, l_1)$
4	$\pi_t^{DL}(v, 4) = d(v, l_1) - d(l_1, t) $
5	$\pi_t^{DL}(v, 5) = d(v, l_2) - d(l_2, t) $
6	$\pi_t^{DL}(v, 6) = \frac{ d(v, l_1) - d(l_1, l_2) \times d(l_1, l_2) - d(l_2, t) - d(v, l_1) \times d(l_2, t)}{d(l_1, l_2)}$

Table 1: Dual Landmark Heuristics for ALP

In practical cases, information regarding the values of $d(A, C)$ and $d(B, D)$ (the diagonals) may be unknown. Therefore, the distance between can be estimated as follows. First, suppose all the values on the right side of Equation 10 are known (except, of course, the distance between vertices A and D) and the values on the left side are unknown. Using the reverse triangle inequality, we understand that

$$1 \leq |d(A, B) - d(B, C)| \leq d(A, C) \quad (12)$$

$$1 \leq |d(B, C) - d(C, D)| \leq d(B, D) \quad (13)$$

for $A \neq B \neq C \neq D$. Because they are non-negative, we also know that

$$1 \leq |d(A, B) - d(B, C)| \times |d(B, C) - d(C, D)| \leq d(A, C) \times d(B, D) \quad (14)$$

Using these lower bounds, we can rewrite Ptolemy's inequality with respect to the lower bound for the distance between vertices A and D as

$$\frac{|d(A, B) - d(B, C)| \times |d(B, C) - d(C, D)| - d(A, B) \times d(C, D)}{d(B, C)} \leq d(A, D) \quad (15)$$

Because we use Ptolemy's inequality here, this can become a perfect estimate when a cyclic quadrilateral is formed from the four endpoint vertices, $A, B, C, D \in V$.

Because multiple points are used, more inequalities can be generated to estimate distances. The maximum over the set of lower bounds derived by these inequalities can be used to tighten the lower bound for the distance between A and D. With that said, two more lower bounds for A and D, derived from the triangle inequality, are noted here:

$$|d(A, B) - d(B, D)| \leq d(A, D) \quad (16)$$

$$|d(A, C) - d(C, D)| \leq d(A, D) \quad (17)$$

3.2 Dual Landmark Heuristics for A*

Just as with triangle inequalities, the lower bound produced by the quadrilateral inequalities in the previous subsection can be used as heuristics for the A* algorithm. By choosing two landmark vertices to act as endpoints B and C, new heuristics are computed as follows: At the visited vertex and target nodes $v, t \in V$ and two valid landmark vertices $l_1, l_2 \in V$ in a graph G, let $\pi_t^{DL}(v, i), i \in [1, 6]$ denote each new heuristic function for A*. Table 1 lists each heuristic based on the mentioned lower bounds. Each of these are new, admissible heuristics for A* based on polygon inequalities, specifically for quadrilaterals. The following is the optimal dual landmark heuristic now proposed for ALP:

$$\pi_t^{DL}(v) = \max_i \pi_t^{DL}(v, i) \quad (18)$$

Just as with ALT, the maximum over the set of available lower bounds is used to tighten the lower bounds. The A* algorithm is used with this new heuristic function as input, just as in ALT, with one change. This change involves a process known as *distributed embedding*. The distributed embedding process is further detailed in a later section. In summary, after landmark selection, each vertex in the graph is assigned to a single landmark. These vertices contain distance information for only the landmark to which they are assigned. As a vertex v is visited, if v does not have distance information at its current landmark node, l_1 , the landmark that does have distance information for v is used to bound the search. For unidirectional A*, the l_2 landmark remains the same for the target node, as it is the only one containing distance information for that node. This fact, of course, would change for the bidirectional variant of A*. Note that, when using distributed embedding, $\pi_t^{DL}(v, 4)$ and $\pi_t^{DL}(v, 5)$ can only be used when both the visited node v and target node t share the same landmark. Otherwise, the information needed for this heuristic cannot be computed. If the source and target vertex share the same landmark (i.e., $l_1 = l_2$), then the ALP heuristic is reduced to the ALT heuristic (i.e., $l_1 = l_2 = l$) as follows:

$$\pi_t^{DL}(v, 1) = |d(v, l) - d(l, l)| - d(l, t) = |d(v, l)| - d(l, t) = d(v, l) - d(l, t) \quad (19)$$

$$\pi_t^{DL}(v, 2) = |d(v, l) - d(l, t)| - d(l, l) = |d(v, l) - d(l, t)| = |d(v, l) - d(l, t)| \quad (20)$$

$$\pi_t^{DL}(v, 3) = |d(l, l) - d(l, t)| - d(v, l) = |-d(l, t)| - d(v, l) = d(l, t) - d(v, l) \quad (21)$$

Because we are taking the maximum, $\pi_t^{DL}(v, 1)$ and $\pi_t^{DL}(v, 3)$ simplify to the reverse triangle inequality. $\pi_t^{DL}(v, 4)$ and $\pi_t^{DL}(v, 5)$ are, by their very definition, equal to the reverse triangle inequality, as well. $\pi_t^{DL}(v, 6)$ cannot be used over the same set of landmarks because its equation would result in a division by zero. Therefore, this dual landmark ALP heuristic always reduces to a triangle inequality heuristic when the visited vertex and the target vertex share a landmark. Note that this does not mean that π_t^{DL} necessarily reduces to the ALT heuristic in this case. For this to be the case, the shared landmark would have to be the landmark that would have maximized π_t^L . Also, we will see later, in Section 5.1, that this ALP heuristic can still computationally outperform the ALT heuristic.

This ALP heuristic function for dual landmarks is characterized in Section 4. It should be noted that there are other quadrilateral inequalities for special cases and shapes that could also be used to define A* heuristics, as they, too, can yield estimates that never overestimate the shortest path. Quadrilaterals maintain these inequalities along with others that take into account perimeter, convexity, area, and a whole host of other geometric qualities that can be mapped to graphs. The examples for the dual landmark ALP heuristic used in this paper, however, are simply to demonstrate the ALP technique's utility in comparison to ALT.

3.3 Distributed Embedding

ALP can exhibit a space complexity of $\theta(|L|^2 + |V|)$ using the following technique, called *distributed embedding*. In the proposed dual landmark preprocessing for ALP, each landmark only computes the SPT for the subgraph induced by the landmark's graph partition. The only other calculation is an all-pairs shortest path calculation among the landmarks. For best results, each partition should be a connected subgraph to increase the likelihood that the shortest path from the landmark to any vertex in the partition lies entirely within the subgraph induced by the graph partition, though this is not a requirement.

An example of this distributed embedding technique is illustrated in Figure 2. During preprocessing, each vertex in the graph is labeled with an identifier, signifying its landmark partition and the distance to its corresponding landmark. Any of the embedding methods mentioned in Section 2.3 can be used for the subgraph induced by the graph partition to select an optimal set. Once all landmarks have been chosen, an SPT for each landmark in L is then computed for its respective partition. This partitioning information is not explicitly stored by an external data structure. Rather, each vertex maintains distance information about the landmark to which it belongs along with a reference to that landmark. For landmark selection algorithms, if

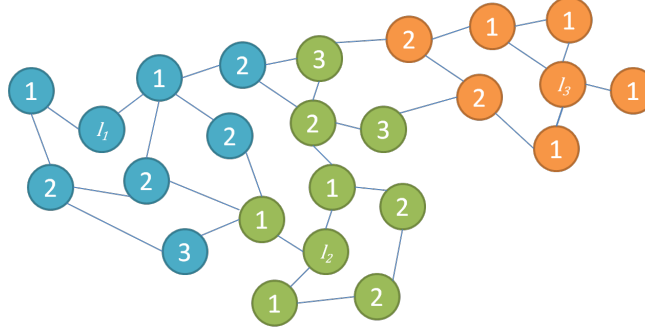


Figure 2: Example of Vertex Labeling for the Dual Landmark ALP Heuristic with Distributed Embedding

an algorithm requires understanding of all vertices that belong to a particular partition, then the partition can be discovered by finding all vertices with a common landmark reference. During query time, ALP carries out the normal A* algorithm with the ALP heuristic function, π_t^{DL} , that relies on polygon inequalities for quadrilaterals.

Here, we describe distributed embedding in the context of the dual landmark approach, which leverages quadrilateral inequalities in the graph. This technique can identify other shapes to draw inequalities from (pentagons, hexagons, heptagons, etc.) by identifying multiple landmarks within a graph partition. For example, if two landmarks are chosen within each subgraph, the data structure contains enough information to leverage geometric inequalities for polygons with up to nine sides (nonagons). Choosing three landmarks within a subgraph would enable the leveraging of inequalities for polygons of up to sixteen sides (hexadecagons). The available inequalities are simply limited to the size of the graph (See full paper for details on how to compute the available inequalities for an embedding).

4 Analysis of Dual Landmark Heuristics

In this section, we provide analysis for the dual landmark heuristic, π_t^{DL} , with respect to A* and ALT. For a source and target vertex pair, the following theorems apply to π_t^{DL} :

Theorem 1 (Admissibility). π_t^{DL} is an admissible heuristic.

Proof. The proofs for the inequalities used for the heuristic are all derived in the previous section. Because the heuristic function has an upper bound set at the actual shortest path to the target, the heuristic will never overestimate the distance to the target, rendering it admissible. \square

Theorem 2 (Non-monotonicity). Using distributed embedding, π_t^{DL} is not consistent.

Proof. Let c be the cost of transitioning with A* from vertex v to v' , for $v, v' \in V$. Recall that c is non-negative for the A* algorithm. Let $\pi_t^{DL}(v, 1)$ be the maximum chosen for π_t^{DL} for both of these iterations. Then, for π_t^{DL} to be consistent,

$$|d(v, l_1) - d(l_1, l_2)| - d(t, l_2) \leq c + |d(v', l_1) - d(l_1, l_2)| - d(t, l_2) \quad (22)$$

Because c is non-negative and the heuristic takes into account whether or not it moves towards or away from its landmark, $d(v', l_1) = d(v, l_1) - c$ or $d(v', l_1) = d(v, l_1) + c$, respectively. Therefore, this equation holds and demonstrates monotonicity over the same set of landmarks for successive iterations. However, allow the selection of landmarks for a query to change during the query, due to distributed embedding. For the

heuristic to be consistent, with vertex v belonging to landmark l_i and v' belonging to a different landmark l_j , once again let $\pi_t^{DL}(v, 1)$ be the maximum chosen for π_t^{DL} for both of these iterations. The following equation must then hold for π_t^{DL} to be consistent:

$$|d(v, l_i) - d(l_i, l_2)| - d(t, l_2) \leq c + |d(v', l_j) - d(l_j, l_2)| - d(t, l_2) \quad (23)$$

Let l_j be chosen such that the $d(l_i, l_2) > c + d(l_j, l_2)$ and $d(v, l_i) < d(v', l_j)$. This scenario yields a contradiction for the above equation. Therefore, π_t^{DL} is not consistent. \square

Theorem 3. π_t^{DL} does not dominate π_t^L over the same set of landmarks.

Proof. In the previous section, we demonstrated that the dual landmark heuristic reduces to the triangle inequality heuristic over the same set of landmarks. The ALP estimate can then be, at most, equal to the estimate of ALT. \square

This means that, at best, the ALP heuristic estimates will be equal to that of ALT's and at worst, will always be less than ALT's.

Theorem 4. π_t^L does not dominate π_t^{DL} over an unequal set of landmarks.

Proof. This can be proven by contradiction. Let $\pi_t^{DL}(v, 1)$ be the maximum chosen value for π_t^{DL} . For the triangle inequality heuristic to dominate the dual landmark heuristic:

$$|d(v, l) - d(t, l)| \geq |d(v, l_i) - d(l_i, l_j)| - d(t, l_j) \quad (24)$$

There are many scenarios that can contradict this statement. One example is when both v and t have equal distance from the ALT landmark and $d(l_i, l_j) > d(v, l_i) + d(t, l_j)$. In this case, the statement does not hold, yielding a contradiction. \square

To summarize, according to Theorem 1, ALP's dual landmark heuristic is an admissible heuristic, making it a viable candidate for the A* algorithm, even though it is not consistent when using distributed embedding, as shown in the proof of Theorem 2. From Theorem 3, this heuristic for ALP does not dominate the heuristic for ALT over the same set of landmarks. From Theorem 4, it is demonstrated that there are scenarios in which the ALP heuristic gives a higher estimation than the ALT algorithm. In the proof for Theorem 4, a possible scenario for ALT (with all landmarks being equidistant to v and t such that the heuristic estimate is equal to 0) is used to theoretically demonstrate that it can have a lower heuristic estimate than ALP. The proof inherently shows the reverse, as well: that ALP can have a lower heuristic estimate than ALT.

4.1 Comparing Landmark Configurations

Recall that one A* heuristic outperforms the other, in terms of the number of vertices that are searched, by creating a higher estimation of the shortest path lower bound. In this section, we use this metric for performance to compare ALP's dual landmark heuristic to the ALT heuristic. Let $l_\alpha \in \mathbf{L}$ be the landmark chosen for ALT that maximizes its heuristic and $l_1, l_2 \in \mathbf{L}$ be the landmarks for the current vertex and the target, respectively.

Scenario 1: $l_1 = l_\alpha \neq l_2$

Outperforms ALT when $|d(l_1, l_2) - d(t, l_2)| - d(v, l_1) \geq |d(v, l_1) - d(t, l_1)|$.

This scenario, in particular, outperforms ALT at the beginning of a search in a large graph, for $\pi_t^{DL}(v, 2)$,

when the distances between the two landmarks is significantly large. Particularly, if $|d(l_1, l_2) - d(t, l_2)| \gg d(t, l_1) > d(v, l_1)$, the ALP heuristic will provide higher estimates than ALT. As such, $\pi_t^{DL}(v, 3)$ and $\pi_t^{DL}(v, 6)$ are the estimates that have a higher likelihood of yielding stronger results than the triangle inequality here.

Scenario 2: $l_1 \neq l_\alpha = l_2$

Outperforms ALT when $|d(l_1, l_2) - d(v, l_1)| - d(t, l_2) \geq |d(v, l_2) - d(t, l_2)|$.

Particularly, if $|d(l_1, l_2) - d(v, l_1)| \gg d(v, l_2) > d(t, l_2)$, the heuristic dominates. Since we cannot rely on $d(v, l_1)$ to always be significantly larger than $d(v, l_2)$, the heuristic relies on the distance between the respective landmarks being significantly large to dominate. Therefore, in this scenario, the ALP heuristic dominates ALT **when the distance between the two landmarks is significantly large.** As such, $\pi_t^{DL}(v, 1)$ and $\pi_t^{DL}(v, 6)$ are the estimates that have a higher likelihood of yielding stronger results than the triangle inequality here.

Scenario 3: $l_1 = l_\alpha = l_2$

Always has the same performance as ALT. $d(l_\alpha, l_\alpha) = 0$, by definition. Therefore, all of the possible equations for the ALP heuristic are reduced to the triangle inequality. And **the ALP heuristic becomes the ALT heuristic.**

Scenario 4: $l_1 = l_2 \neq l_\alpha$

Outperforms ALT when $|d(v, l_1) - d(l_1, t)| - d(l_1, l_1) \geq |d(v, l_\alpha) - d(t, l_\alpha)|$.

$d(l_1, l_1) = 0$, by definition. Therefore, $\pi_t^{DL}(v, 6)$ is eliminated as an option for the dual landmark heuristic. Because this occurs and because the ALT heuristic chooses the landmark that maximizes the triangle inequality, the best we can hope for is that the ALP heuristic is reduced to the heuristic for ALT. Therefore, **when the ALP algorithm's search is in the same partition, the ALP algorithm never dominates the ALT algorithm.**

Scenario 5: $l_1 \neq l_\alpha \neq l_2$

Outperforms ALT when

$$\pi_t^{DL}(v, 1) \geq |d(v, l_\alpha) - d(t, l_\alpha)| \text{ or } \pi_t^{DL}(v, 2) \geq |d(v, l_\alpha) - d(t, l_\alpha)| \text{ or}$$

$$\pi_t^{DL}(v, 3) \geq |d(v, l_\alpha) - d(t, l_\alpha)| \text{ or } \pi_t^{DL}(v, 6) \geq |d(v, l_\alpha) - d(t, l_\alpha)|.$$

$\pi_t^{DL}(v, 4)$ or $\pi_t^{DL}(v, 5)$ can only reach the equivalence of the ALT heuristic's estimate over the same set of landmarks or for landmarks with similar distances to the one's used in ALT. Scenario 5 is the most common situational scenario. This is also the scenario that most significantly demonstrates that when the landmarks for ALT and ALP differ, the heuristic value for ALP is not always greater than the heuristic value for ALT, producing the results of Theorems 3 and 4.

5 Pathfinding Performance

5.1 Measuring A* Heuristic Performance

Thus far, when describing ALP's performance in comparison to ALT, performance has been measured by the value calculated by a heuristic function. For A*, this value determines the size of the search space for any given query. However, one thing that is not taken into account in this and the proposal of other shortest path heuristics is the amount of processing needed to compute the actual heuristic at each visited vertex. In ALT, for each PPSP query, at each vertex, a number of subtractions equal to the number of landmarks is performed as well as a max operation. This means a $\theta(|L|)$ runtime for every visited node. For large-scale graphs, which require more landmarks to be preprocessed, this can significantly add to the overall compute time of queries. In comparison, with the proposed dual landmark ALP heuristic, if the visited vertex and the target vertex are owned by different landmarks, exactly nine subtraction operations, two multiplication operations, and a single division operation occurs with an $O(4)$ max operation. Over the same set of landmarks, dual landmark ALP executes eight subtraction operations and an $O(5)$ max operation

upon visiting a vertex. This means that, in terms of practical, processor-based performance measurements, even over the same set of landmarks, it is possible for dual landmark ALP to outperform ALT. Because Dijkstra’s algorithm performs no arithmetic operations at each visited vertex, it is possible for Dijkstra’s algorithm to outperform both ALT and ALP algorithms at large scales (See Section 5.1 in full paper).

5.2 Computational Complexity

The benefits in time and space complexity of ALP over ALT is due to the novelty of distributed embedding. Note that the worst-case time complexity for preprocessing of ALP remains the same as that of ALT. The actual shortest path between two vertices within a graph partition could include vertices from outside the partition. This means that, in the worst case, the generated SPT for a landmark within a subgraph includes the entire vertex set of the graph. This phenomena would rarely happen in practice. In practice, the SPT will be significantly small in comparison to the size of the graph and its generation will run in a fraction of the time. Therefore, for a graph in which the vertices of each partition match the vertices in a partition’s shortest path tree, let E' be the average number of edges in each partition and V' the average number of vertices in each partition. Then the average runtime of ALP preprocessing, not including landmark selection, is

$$\theta(|L| \cdot (|E'| + |V'| \log |V'|)) \quad (25)$$

Because the shortest path tree is computed from every chosen landmark and distance along with an all-pairs shortest path calculation for the landmarks, ALT’s data structure requires $\theta(|L| \cdot |V| + |L|^2)$ space, where L is the set of chosen landmarks. Since $|L| \leq |V|$, the theoretical space requirement for ALT can be said to be $O(|V|^2)$. Note that this upper limit is only theoretical, as a relatively small number of landmarks are chosen for any particular graph. Therefore, the $\theta(|L| \cdot |V| + |L|^2)$ space requirement is a more practical specification. For ALP, shortest path data is stored for the landmark-vertex pairs of each graph partition and the pairwise distances between landmarks. Therefore, ALP’s data structure requires $\theta(|V| + |L|^2)$ space. Once again, because $|L| \leq |V|$, the space requirement for ALP can also be described as $O(|V| + |V|^2) = O(|V|^2)$, which is theoretically larger than the worst-case ALT requirement.

6 Conclusions

In this paper, we identify a heuristic for A* that leverages a data structure of size $\theta(|L|^2 + |V|)$ as opposed to ALT’s previous $\theta(|L| \cdot |V|)$. This data structure is formed through a new embedding process, which requires identifying distances only within a graph partition. We’ve also identified each theoretical scenario in which ALP’s estimates are greater than ALT’s. Finally, we have established that in cases in which the ALT heuristic has greater average estimates than the ALP dual landmark heuristic, ALP can still computationally outperform ALT and Dijkstra’s algorithm can outperform A* using either preprocessing technique. The fact that Dijkstra’s algorithm can outperform both of these methods as graphs scale should serve as a cautionary example for other methods of shortest path preprocessing.

7 Acknowledgments

Special thanks to advisor Dr. Michael J. Laszlo and my doctoral committee at the Nova Southeastern University GSCIS for their support throughout the Computer Science PhD program. Also, special thanks to my employer, Raytheon BBN Technologies, for mentorship, guidance, and financial aid throughout the course of the CISD program.

References

- [1] A. Awasthi, Y. Lechevallier, M. Parent, and J. M. Proth. Rule based prediction of fastest paths on urban networks. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 978–983, 2005.
- [2] D. Delling, P. Sanders, D. Schultes, and D. Wagner. *Highway Hierarchies Star*, volume 74 of *The Shortest Path Problem: Ninth Dimacs Implementation Challenge (Dimacs Series in Discrete Mathematics and Theoretical Computer Science)*. American Mathematical Soc., 2009.
- [3] D. Delling and D. Wagner. Landmark-based routing in dynamic graphs, 2007.
- [4] R. Duan, S. Pettie, and Siam/Acm. Dual-failure distance and connectivity oracles. *Proceedings of the Twentieth Annual Acm-Siam Symposium on Discrete Algorithms*, pages 506–515, 2009.
- [5] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Graph Theory 1736-1936*, 1736.
- [6] A. Goldberg, H. Kaplan, and R. F. Werneck. Reach for a*: Efficient point-to-point shortest path algorithms. In *SIAM Workshop on Algorithms Engineering and Experimentation (ALENEX 06)*, number MSR-TR-2005-132, page 41, Miami, FL, January 2006. Society for Industrial and Applied Mathematics. also as MSR-TR-2005-132.
- [7] A. V. Goldberg and C. Harrelson. Computing the shortest path: A* search meets graph theory. *Proceedings of the Sixteenth Annual Acm-Siam Symposium on Discrete Algorithms*, pages 156–165, 2005.
- [8] A. V. Goldberg and R. F. Werneck. Computing point-to-point shortest paths from external memory. In *Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX05)*, pages 26–40, 2005.
- [9] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. *Proximity search in databases*, pages 26–3737. Morgan Kaufmann Publishers Inc, 1998.
- [10] R. Gutman. Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 100–111. SIAM, 2004.
- [11] D. Kay. *College Geometry: A Unified Development*. Taylor & Francis, 2011.
- [12] R. H. M, 246, hring, H. Schilling, B. Sch, 252, tz, D. Wagner, and T. Willhalm. Partitioning graphs to speedup dijkstra’s algorithm. *J. Exp. Algorithmics*, 11:2.8, 2007.
- [13] J. Maue, P. Sanders, and D. Matijevic. Goal-directed shortest-path queries using precomputed cluster distances. *J. Exp. Algorithmics*, 14:3.2–3.27, 2010.
- [14] J. Maue, P. Sanders, D. Matijevic, C. Alvarez, and M. Serna. Goal directed shortest path queries using precomputed cluster distances. *Experimental Algorithms, Proceedings*, 4007:316–327, 2006.
- [15] P. Sanders and D. Schultes. Engineering fast route planning algorithms. *Experimental Algorithms, Proceedings*, 4525:23–36, 2007.
- [16] J. Sankaranarayanan and H. Samet. Query processing using distance oracles for spatial networks. *Knowledge and Data Engineering, IEEE Transactions on*, 22(8):1158–1175, 2010.
- [17] M. Thorup and U. Zwick. Approximate distance oracles, 2001.

- [18] D. Wagner, T. Willhalm, and C. Zaroliagis. Geometric containers for efficient shortest-path computation. *J. Exp. Algorithmics*, 10:1.3, 2005.
- [19] S. Yussof, R. A. Razali, S. Ong Hang, A. A. Ghapar, and M. M. Din. A coarse-grained parallel genetic algorithm with migration for shortest path routing problem. In *High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on*, pages 615–621, 2009.
- [20] A. A. A. Zakzouk, H. M. Zaher, and R. A. Z. El-Deen. An ant colony optimization approach for solving shortest path problem with fuzzy constraints. In *Informatics and Systems (INFOS), 2010 The 7th International Conference on*, pages 1–8, 2010.
- [21] X. Zongyan, L. Haihua, and G. Ye. A study on the shortest path problem based on improved genetic algorithm. In *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, pages 325–328, 2012.